Webbasierte Anwendungen SoSe 2018
Übungsaufgaben - ECMA 6 Basics

Die folgenden Aufgaben sollen Ihnen die Arbeit mit externen Bibliotheken in der Projektphase erleichtern. Daher zeigen wir Ihnen einige ausgewählte ECMA 6 Features, einer „neueren“ Version von Javascript, um die Einarbeitungsphase zu verkürzen. Diese Features werden Ihnen in diversen Bibliotheken auf Github begegnen.

Scope

Neben dem Schlüsselwort var existieren seit ECMA 6 zwei weitere Möglichkeiten Variablen zu initialisieren: let und const. Das Schlüsselwort let verhält sich genauso wie var, allerdings ist die Sichtbarkeit auf den umfassenden Block limitiert. Das folgende Beispiel macht dies deutlich, in dem eine Variable definiert wird, welche nur innerhalb der Schleife existiert. Da Variablen mit dem Schlüsselwort var global sind beugt let durch die blockbasierte Sichtbarkeit mögliche Fehler vor.

```
for(var x = 0; x < 10; x++) {
	console.log(x); // x ist 0-9
}
console.log(x); // x ist immer noch vorhanden (x = 10)

for(let z = 0; z < 10 ; z++) {
	console.log(z); // z ist 0-9
}
console.log(z); // ReferenceError
```

In dem Beispiel wäre die Variable x nach dem Durchlaufen der Schleife immer noch mit dem Wert 10 vorhanden. Das Beispiel mit dem Schlüsselwort let erzeugt einen ReferenceError, da die Variable z nicht mehr existiert. Variablen mit dem Schlüsselwort const verhalten sich ähnlich wie let, allerdings mit dem wichtigen Unterschied das diese nach der Instanziierung nicht mehr geändert werden können.

Arrow Functions

In ECMA 6 existiert außerdem eine andere Schreibweise für Funktionen. Sie ist nicht nur kürzer sondern übernimmt auch das this einer umgebenden Funktion. So ist es nicht mehr nötig this in einer temporären Variable zu speichern, wenn man diese innerhalb eines Callbacks verwenden möchte. Dies bezeichnet man auch als „lexical this“.

```
// Alte Schreibweise
function() {
	var self = this;
		this.array.forEach(function(value) {
		if(value % 2 === 0)
			self.selection.push(value);
	});
}
```

```
// Neue Schreibweise
() => {
	this.array.forEach((v)  => {
		if(v % 2 === 0)
			this.selection.push(v);
	});
}
```

Template Literals

Ein weiterer Unterschied ist die Verwendung von String Interpolation, auch bekannt als Template Literals. In früheren Versionen von Javascript wurde die Konkatenation mit Hilfe des Plus-Zeichens verwendet, welche auch heute noch funktioniert. Ein wichtiger Unterschied ist dabei, dass man dafür nicht mehr die üblichen Anführungszeichen verwendet, sondern Backticks.

```
const port = 3000;

server.listen(port, () => {
	console.log(`Server online at port ${port}`);
	// Server online at port 3000
});
```

Destructuring

Mit Hilfe des Destructuring lassen sich Elemente aus Objekten oder Arrays direkt Variablen zuweisen, ohne diese Elemente einzeln zuweisen zu müssen.

```
// Array Destructuring
let list = [1, 2, 3];
let [a, b] = list;

console.log(a, b) // output: 1 2

// Object Destructuring
let person = {firstname: ”Max”, lastname: ”Mustermann”};
const {firstname, lastname} = person;

console.log(firstname, lastname); // Max Mustermann
```

Aufgabe 1)

Das folgende Beispiel zeigt einen Ausschnitt Ihres Programmcodes während der Projektphase.

```
for(var x = 0; x < 10; x++) {
        console.log(x);
        setTimeout(function() {
                console.log('The number is ' + x);
        }, 1000);
}
```

Überlegen Sie welche Werte im zweiten console.log() ausgegeben werden. Im Anschluss führen Sie diese Funktion in Node.js aus. Gibt es Abweichungen zu Ihren Werten? Falls ja, wodurch könnte dies entstehen und wie könnten Sie diese Abweichung beheben?

Aufgabe 2)

Schreiben Sie die folgende Funktion so um, dass sie die hier vorgestellten Neuerungen verwendet.

```
function getTweets( id ) {
        return fetch("https://api.twitter.com/user/" + id )
                .then(function(response) {
                        return JSON.parse(response);
                }).then( function(response) {
                        return response.data;
                }).then(function(tweets) {
                        return tweets.filter(function(tweet) {
                                return tweet.stars > 50
                        })
                }).then(function(tweets) {
                        return tweets.filter(function(tweet) {
                                return tweet.rts > 50
                        })
                })
}
```

Aufgabe 3)

Schreiben Sie eine Funktion, welche ein Array von zwei ganzen Zahlen entgegen nimmt und die Elemente sortiert, ohne das die zu tauschenden Werte in einer zusätzlichen Variable gespeichert werden.

```
// Beispiel
let array = [4, 1];

console.log(sortInteger(array));
```